<u>Remarks/Arguments</u>

With reference to the Office Action mailed May 23, 2007, Applicants offer the following remarks and argument.

Status of the Claims

Claims 1-27 were originally presented for examination. Claims 1, 6-7, 9, 11, 13, 20, and 25-26 have been amended. Claims 4-5, 10, 12, 14-19, and 23-24 have been canceled.

Claim Numbers	Status
1	Amended
2-3	Original
4-5	Canceled
6-7	Amended
8	Original
9	Amended
10	Canceled
11	Amended
12	Canceled
13	Amended
14-19	Canceled
20	Amended
23-24	Canceled
25-26	Amended
27	Original

Thirteen claims have been canceled, being a reduction of 48 percent in the number of claims.

The Art of Record

The primary reference, Published United States Patent Application 2003/0081792 of Nakano et al. for Digital Work Protection System, Key Management Apparatus, And User Apparatus describes a digital work protection system in which a user apparatus can efficiently determine a key assigned to the user apparatus. Nakano's system has (a) either (i) a recording apparatus that records digitized content such as a movie, or (ii) a reproduction apparatus that reproduces the digitized content, and (b) a recording medium. A media key for use in recording or reproduction is encrypted by a plurality of device keys and recorded on the recording medium. The recording apparatus or the reproduction apparatus specifies the encrypted media key that it is to decrypt, from among a plurality of encrypted media keys. A key management apparatus records node revocation patterns assigned to nodes in a tree structure to the recording medium in a particular order, as header information of key information, together with the encrypted media keys. The recording apparatus or the reproduction apparatus specifies the encrypted media key to be decrypted, by analyzing the node revocation patterns sequentially.

Especially relied upon in the Office Action were paragraphs [0030] and [0032], i.e.,

> [0030] Here, in the key management apparatus, the n-ary tree may be composed of a plurality of layers, the order in which key information generation unit writes the plurality of encrypted media keys to the recording medium may be an order of the layers from a root-side layer to a leaf-side layer, the root being a starting point, and the revocation information generation unit may write the pieces of revocation information to the recording medium in the order. In the user apparatus, the n-ary tree may be composed of a plurality of layers, the order in which the plurality of encrypted media keys are written to the recording medium may be an order of the layers from a root-side layer to a leaf-side layer, the root being a starting point, the pieces of revocation information may be written to the recording medium in the order, and the specification unit may specify the encrypted media key from amongst the plurality of encrypted media keys written in the order, with use of the plurality of pieces of revocation information written in the order.

and

> [0032] Here, the order in which the key information generation unit writes the plurality of encrypted media keys to the recording medium may be an order in which the nodes are positioned on the paths from the root to the leaves, the root being a starting point and each node being included only once in the order, and the revocation information generation unit may write the pieces of revocation information to the recording medium in the order. In the user apparatus, the n-ary tree may be composed of a plurality of layers, the order in which the plurality of

> encrypted media keys are written to the recording medium may be an order in which the nodes are positioned on the paths from the root to the leaves, the root being a starting point, and each node being included only once in the order, the pieces of revocation information may be written to the recording medium in the order, and the specification unit may specify the encrypted media key from amongst the plurality of encrypted media keys written in the order, with use of the plurality of pieces of revocation information written in the order.

The secondary reference, United States Patent 7,095,850 to McGrew for Encryption Method And Apparatus With Forward Secrecy And Random Access Key Updating Method describes an encryption method and apparatus that provides forward secrecy. It accomplishes this by updating the key using a one way function after each encryption. By providing forward secrecy within a cipher, rather than through a key management system, forward secrecy may be added to cryptographic systems and protocols by using the cipher within an existing framework. A random access key updating method can efficiently generate one or more future keys in any order. McGrew et al disclose that embodiments are applicable to forward secret ciphers that are used to protect protocols with unreliable transport, to ciphers that are used in multicast or other group settings, and to protection of packets using the IPSec protocols.

Column 5, lines 43-48, of McGrew et al.,

> A random access key updating method can efficiently generate one or more future keys in any order. Embodiments are applicable to forward secret ciphers that are used to protect protocols with unreliable transport, and are applicable to ciphers that are used in multicast or other group settings.

was cited to overcome conceded deficiencies of the primary reference, Nakano et al.

Discussion

The Office Action and the claims will be discussed with respect to Claim 1, since claim 1 is exemplary of the other independent claims, and all of the pending claims are either dependent on clam 1 or on claims that are substantially parallel to claim 1.

Applicants' invention relates to hierarchal keys, and, generally, to generating hierarchical

keys of digital assets. In the hierarchical key generating method of the claimed invention, all the assets are managed in a tree structure. The tree structure comprises group nodes and leaf nodes, each group node including a root node and other group nodes being non-root nodes, and each node corresponds to different collections in the digital assets. Each higher level node is called a father node of its lower node, and each lower node is called a child node of its higher node. In applicant's claimed invention, only the key of the root node is given randomly, while the keys of other nodes are obtained by computing by using a one-way function according to the key of the father nodes of the nodes.

Applicants' claimed invention further includes encrypting the digital assets on the digital asset server, and subsequently serving requested encryption keys and requested encrypted assets to properly authenticated users.

This is recited in the independent claim as:

> 1. (Currently Amended) A method for generating hierarchical keys of digital assets , encrypting the digital assets in a digital asset server, and utilizing the keys of the digital assets and the encrypted digital assets in an associated digital asset client, comprising the steps of:
>
> arranging the digital assets in the digital asset server as at least one tree structure, a root node of the tree structure representing a complete set of the digital assets, other group nodes representing sub-sets in each level of the digital assets respectively, and the nodes in the lowest level being leaf nodes;
>
> randomly generating ~~the~~ a key of the root node in the digital asset server; and
>
> starting with the key of the root node, using the key of a father node to compute level by level ~~the~~ computed keys of its child nodes through to leaf nodes using a one way function, in the digital asset server [[.]]
>
> encrypting corresponding digital assets in the digital asset server using the computed keys;

requesting an encrypted digital asset at the digital asset client, and determining if a key for the requested encrypted digital asset is present on the digital asset client;

if the digital asset key is not present on the digital asset client, the digital asset client requesting the digital asset key from the digital asset server;

the digital asset server receiver requests from the digital asset client, and thereafter transmitting a digital asset key, if requested, and a requested encrypted digital asset from the digital asset server to the associated digital asset client; and

receiving the key and the encrypted digital asset from the digital asset server at the digital asset client and decrypting the encrypted digital asset utilizing the key.

And is illustrated in Figures 2, 3, and 4

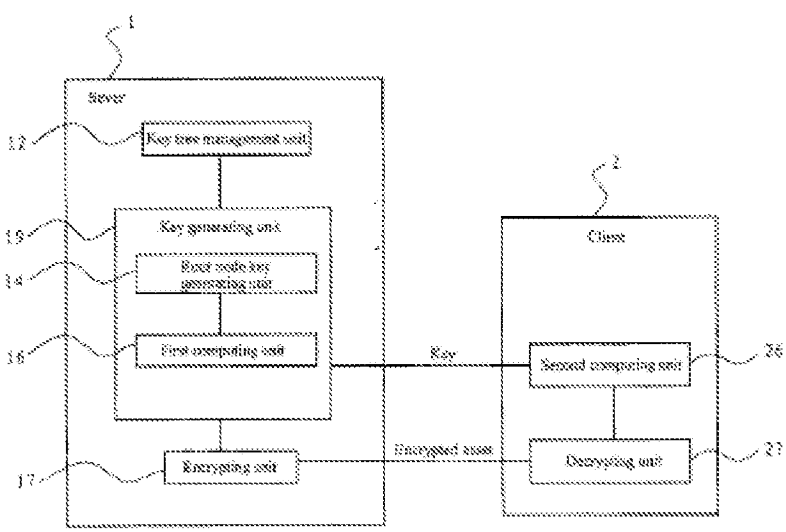

FIG. 2

and described in the Published Application at numbered paragraphs [0034] to [0042][1],

---

1 [0034] FIG. 2 is a schematic diagram of a system embodying the present invention.

[0035] The system in FIG. 2 can be for example an e-learning system (of course the systems of other usage can also be suitable) comprising a server 1 and a client 2 connected with each other. Of course the server 1 and the client 2 can also be connected with each other through such a network (not shown) as the Internet, an enterprise network or an LAN.

[0036] In addition to such general units as a CPU, a bus and a memory (ROM, RAM etc.) (not shown), the server 1 also comprises a key generation unit 19 for generating needed keys. The key generation unit 19 comprises a root node key generating unit 14 for generating a root node key randomly or by other means; and a first computing unit 16 for using a function such as an one-way function to compute level by level the keys of various sub-nodes of the root node, through to leaf nodes, based on the root node key. In the meantime, the server 1 also comprises an encrypting unit 17 for using the computed key to encrypt the assets in corresponding nodes, such as teaching lectures and etc. (in addition, a cipher key is assigned in advance to an asset node such as "math/geometry/rectangular/introduction" randomly or by other means, then the cipher key assigned in advance is encrypted by using the result derived from performing calculation on the computed key using all or a part of the computed keys); and a key tree management unit 12 for creating a key tree structure, maintaining the created tree structure, and storing and managing the computed keys.

[0037] In addition, the sever 1 also comprises a first communication port (not shown) for receiving a request from the client 2 through a corresponding connection line or a network, and transmitting the requested contents and keys, etc. to the client 2.

[0038] In addition to such general units as CPUs, bus and memory (ROM, RAM etc.) (not shown), the client 2 also comprises a second computing unit 26 for using an algorithm program received from the server 1 or embedded internally to compute the keys of needed sub-nodes level by level based on the node keys received from the server 1, so as to compute finally the key of the needed asset node; and a decrypting unit 27 for decrypting the needed assets such as teaching lectures by using the computed keys of the nodes.

[0039] In addition, the client 2 also comprises a second communication port (not shown) for transmitting a request to the server 1 through a corresponding connection line or a network, and receiving the requested contents and keys, etc. from the server 1.

[0040] Of course, in the communication between the above server 1 and client 2, the encrypted assets and the node key needed by the client 2 can be transmitted simultaneously by the server 1 to the client 2, or can be transmitted separately. For example, in the present invention, the server 1 only transmits a corresponding node key to the client 2 upon its request, the encrypted assets and etc. can be transmitted to the client 2 at other time by other means, e.g., the encrypted assets can be transmitted to the client 2 via an optical disc, etc.

[0041] The above first computing unit 16 and second computing unit 26 can be implemented by executing a corresponding software program on the CPU, or can be implemented by a hardware obtained by solidifying a certain software program into the unit such as a CPU. In addition, the root node key generating unit 14, the first computing unit 16 and the encrypting unit 17 in the server 1 are not limited to the above mentioned configuration, but can be implemented by only one unit integrating their functions together. The root node key generating unit 14 and the first computing unit 16 therein are also not limited to the above-mentioned configuration, and they may not be contained in the key generation unit 19, but can be independent from the key generation unit 19. The second computing unit 26 and the decrypting unit 27 in the client 2 are not limited to the above-mentioned configuration, but can be implemented by only one unit integrating their functions together.

[0042] In addition, the above units are not limited to only being contained in the server 1 and the client 2

while the method is illustrated in Figures 3 and 4, and generally at paragraphs [0044]-[0047], [0052], and [0058]-[0066].

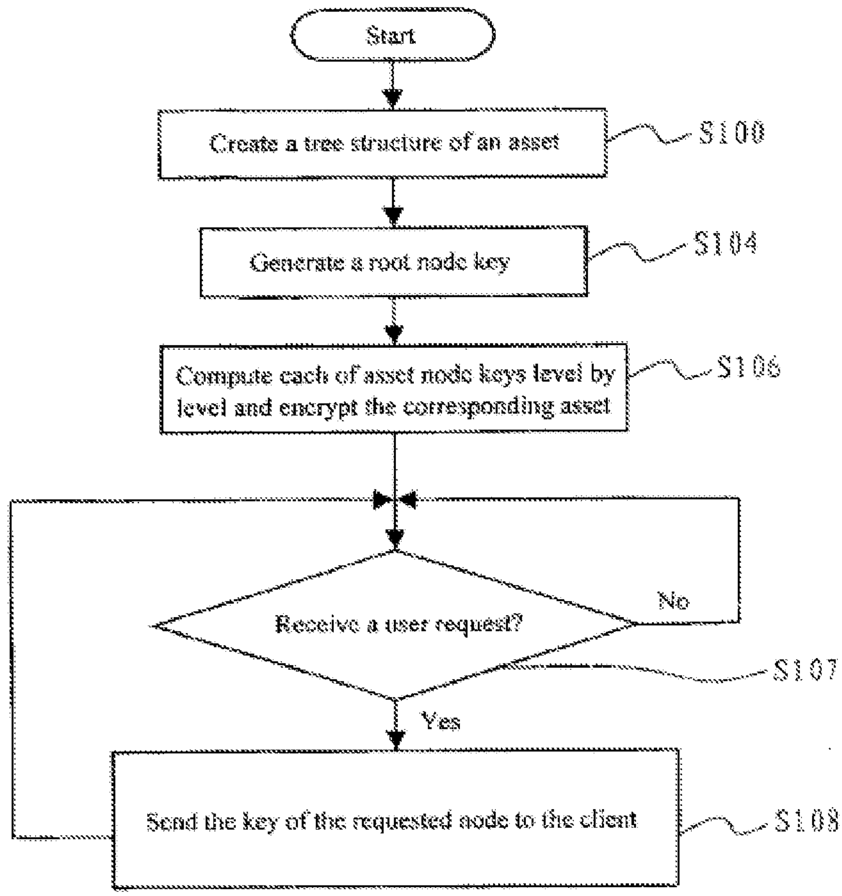

FIG. 3

respectively, but can be arranged outside the server 1 or the client 2 in an operable connection manner.

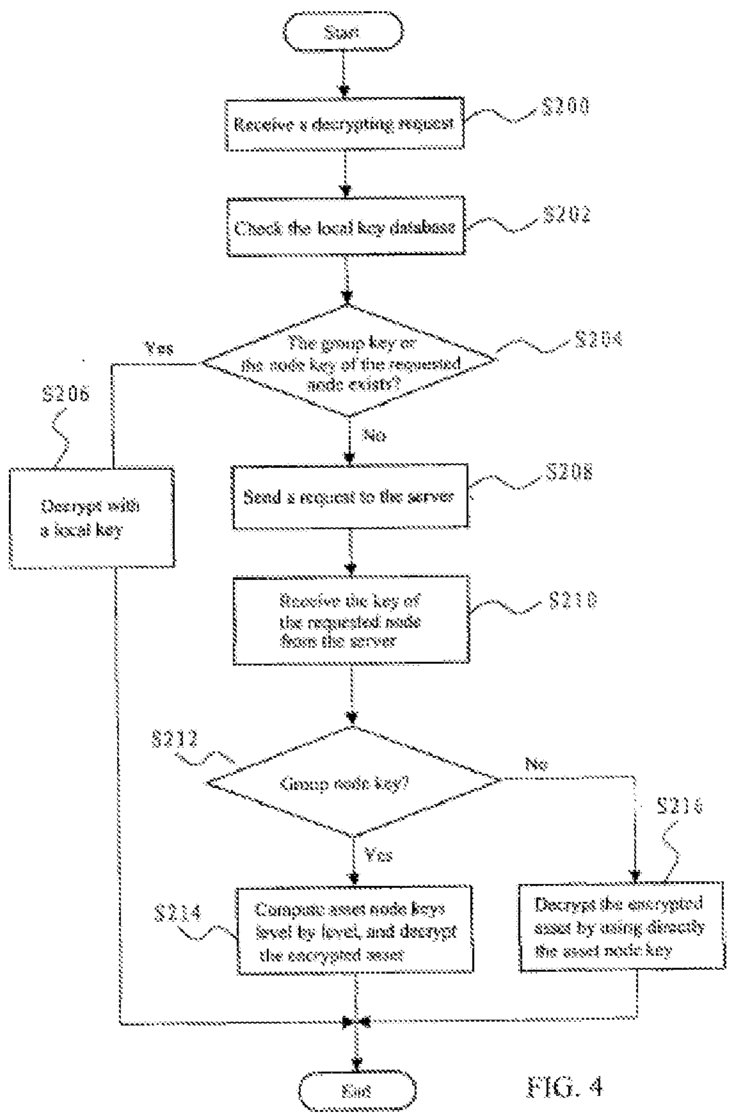

FIG. 4

The amendments to the claims are fully supported by the specification.

<u>The combination of Nakano with McGrew does not render the claimed invention obvious.</u>

It is stated in the Office Action that Nakano teaches the invention recited in original Claim 1 thusly:

A method for generating hierarchical keys of digital assets, comprising the steps of:

arranging the digital assets as at least one tree structure, a root node of the tree structure representing a complete set of the digital assets, other group nodes representing sub-sets in each level of the digital assets respectively, and the nodes in the lowest level being leaf nodes;

is said to be disclosed at paragraphs [0104]-[0106][2], [0302]-[0307][3], and [0417]-[0433][4]

---

2 **ARRANGING THE DIGITAL ASSETS AS AT LEAST ONE TREE STRUCTURE, A ROOT NODE OF THE TREE STRUCTURE REPRESENTING A COMPLETE SET OF THE DIGITAL ASSETS, OTHER GROUP NODES REPRESENTING SUB-SETS IN EACH LEVEL OF THE DIGITAL ASSETS RESPECTIVELY, AND THE NODES IN THE LOWEST LEVEL BEING LEAF NODES;**

[0104] The tree structure T100, as shown in FIG. 4, is a binary tree that has five layers: layer 0 through to layer 4. Since the tree structure T100 is a binary tree, each node (excluding leaves) in the tree structure T100 is connected to two nodes on the lower side of the node via two paths. One node, which is the root, is included in layer 0, two nodes are included in layer 1, four nodes are included in layer 2, eight nodes are included in layer 3, and 16 nodes, which are leaves, are included in layer 4. Note that "lower side" refers to the leaf side of the tree structure, while "upper side" refers to the root side of the tree structure.

[0105] Each of the two paths that connect a node (excluding leaves) in the tree structure T100 with its directly subordinate node is assigned a number, the left path being assigned "0" and the right path being assigned "1". Here, in FIG. 4 a path that branches downwards to the left of a node to connect left nodes is called a left path. A path that branches downwards to the right of a node to connect right nodes is called a right path.

[0106] A node name is assigned to each node. The name of the root node is "root". Each of the nodes in the layers from layer 1 downwards is given a character string as a node name. The number of characters in the character string is equal to the number of the layer, and is generated by arranging the numbers assigned to each node on the same path as the node from the root through to the node in this order. For example, the node names of the two nodes in layer 1 are "0" and "1" respectively. The node names of the four nodes in layer 2 are "00", "01", "10", and "11" respectively. The node names of the eight nodes in layer 3 are "000", "001", "010", "011", . . . , "101", "110" and "111" respectively. The node names of the eight nodes on layer 4 are "0000", "0001", "0010", "0011", . . . , "1100", "1101", "1110", and "1111" respectively.

3 **ARRANGING THE DIGITAL ASSETS AS AT LEAST ONE TREE STRUCTURE, A ROOT NODE OF THE TREE STRUCTURE REPRESENTING A COMPLETE SET OF THE DIGITAL ASSETS, OTHER GROUP NODES REPRESENTING SUB-SETS IN EACH LEVEL OF THE DIGITAL ASSETS RESPECTIVELY, AND THE NODES IN THE LOWEST LEVEL BEING LEAF NODES;**

[0302] The tree structure storage unit 102 has, as one example, a tree structure table D400 shown in FIG. 20 instead of the tree structure table D110.

[0303] The tree structure table D400 corresponds to a tree structure T400 shown in FIG. 19 as one example, and is a data structure for expressing the tree structure T400.

[0304] The tree structure table D400 includes a number of pieces of node information that is equal to the number of nodes in the tree structure T400. The pieces of node information correspond respectively to the nodes in the tree structure T400.

[0305] Each piece of node information includes a node name, a device key, a revocation flag and an NRP.

[0306] The node names, device keys and revocation flags are as described in the first embodiment, therefore descriptions thereof are omitted here.

[0307] The NRP is composed of three bits. The highest bit shows, as described above, that all the user apparatuses assigned to the descendant nodes shown by the corresponding node name are revoked apparatuses. The content of the lower two bits is the same as the NRPs described in the first embodiment.

4

ARRANGING THE DIGITAL ASSETS AS AT LEAST ONE TREE STRUCTURE, A ROOT NODE OF THE TREE STRUCTURE REPRESENTING A COMPLETE SET OF THE DIGITAL ASSETS, OTHER GROUP NODES REPRESENTING SUB-SETS IN EACH LEVEL OF THE DIGITAL ASSETS RESPECTIVELY, AND THE NODES IN THE LOWEST LEVEL BEING LEAF NODES;

[0417] Specifically, the tree structure storage unit 102 is composed of a hard disk unit, and, as shown in FIG. 37, has a tree structure table D1000 shown in FIG. 37 as one example.

[0418] The tree structure table D1000 corresponds to a tree structure T600 shown in FIG. 36 as one example, and is a data structure for expressing the tree structure T600. As is described later, the data structure for expressing the tree structure T600 is generated by the tree structure construction unit 101 as the tree structure table D1000, and written to the tree structure storage unit 102.

[0419] <Tree Structure T600>

[0420] The tree structure T600, as shown in FIG. 36, is a binary tree that has five layers: layer 0 through to layer 4.

[0421] The number of nodes included in each layer is the same as the tree structure T100. Furthermore, the numbers assigned to the paths from the node on the upper side through to the nodes on the lower side are the same as in the tree structure T100. Nodes marked with a cross (X) are revoked nodes.

[0422] The node name of the node that is the root of the tree structure T600 is blank. The node names of the other nodes are the same as in the tree structure T100.

[0423] Each node name is a four-digit expression. The node name of the node that is the root is four blanks. A node name "0" is specifically the character "0"+one blank+one blank+one blank. A node name "00" is the character "0"+the character "0"+one blank+one blank. A node name "101" is the character "1"+the character "0"+the character "1"+one blank. The node name "1111" is the character "1"+the character "1"+the character "1"+the character "1". The other node names are formed similarly.

[0424] In the tree structure T600, "{10}" and the like near each node show NRPs. Furthermore, numbers in circles near each node show the order in which the NRPs are output.

[0425] <Tree Structure Table D1000>

[0426] The tree structure table D1100 includes a number of pieces of node information equal to the number of nodes in the tree structure T1000. Each piece of node information corresponds to one of the nodes in the tree structure T1000.

[0427] Each piece of node information includes a device key and a revocation flag. Node names, device keys and revocation flags are the same as in the tree structure table D100, therefore a description thereof is omitted here.

[0428] Each piece of node information is stored in the tree structure table D1100 in an order shown by the following Order Rule 2. This Order Rule 2 is applied when node information is read sequentially from the tree structure table D1000 by the recording apparatuses 300a etc. and the reproduction apparatuses 400a etc.

[0429] (a) The piece of node information corresponding to the node that is the root is stored at the top of the tree structure table D1000.

[0430] (b) After a piece of node information corresponding to a particular node is stored in the tree structure table D1000, when the node has two subordinate nodes, the node information is arranged in the following manner. Pieces of node information that respectively correspond to each of the left node of the two subordinate nodes and all the further subordinate left nodes on the same path are stored. Then, pieces of node information that respectively correspond to the right node of the two subordinate nodes and all the further right nodes subordinate to the right node are stored.

[0431] (c) Within (b), (b) is re-applied.

[0432] Specifically, the pieces of node information in the tree structure table D1000 shown in FIG. 37 are stored in the following order:

[0433] blank (showing the root), "0", "00", "000", "0000", "0001", "001", "0010", "0011", "01", "010", . . . , "11", "110", "1100", "1101", "111", "1110", and "1111".

Numbered paragraphs [0104]-[0106] describe the topology of the nodes, but no other aspect of claim 1, while numbered paragraphs [0301]-[0304] describe an associated data structure, while numbered paragraphs [0417]-[0433] describe the internal representation of the tree structure and the associated data structure. Specifically, numbered paragraphs [0417]-[0433] describe the tree structure storage unit, and the tree structure table, where the tree structure is again also characterized as a data structure, with specific examples of the data structure. The cited excerpt describes that after a piece of node information corresponding to a particular node is stored in the tree structure table.

generating the key of the root node; and

The bare generation of the key of the root node is said to be taught by Nakano et al. numbered paragraphs [0030][5], [0032][6], [0171-0182][7], and [0248][8].

---

5 **GENERATING THE KEY OF THE ROOT NODE; STARTING WITH THE KEY OF THE ROOT NODE, USING THE KEY OF A FATHER NODE TO COMPUTE LEVEL BY LEVEL THE KEYS OF ITS CHILD NODES THROUGH TO LEAF NODES.**

[0030] Here, in the key management apparatus, the n-ary tree may be composed of a plurality of layers, the order in which key information generation unit writes the plurality of encrypted media keys to the recording medium may be an order of the layers from a root-side layer to a leaf-side layer, the root being a starting point, and the revocation information generation unit may write the pieces of revocation information to the recording medium in the order. In the user apparatus, the n-ary tree may be composed of a plurality of layers, the order in which the plurality of encrypted media keys are written to the recording medium may be an order of the layers from a root-side layer to a leaf-side layer, the root being a starting point, the pieces of revocation information may be written to the recording medium in the order, and the specification unit may specify the encrypted media key from amongst the plurality of encrypted media keys written in the order, with use of the plurality of pieces of revocation information written in the order.

6 **GENERATING THE KEY OF THE ROOT NODE STARTING WITH THE KEY OF THE ROOT NODE, USING THE KEY OF A FATHER NODE TO COMPUTE LEVEL BY LEVEL THE KEYS OF ITS CHILD NODES THROUGH TO LEAF NODES.**

[0032] Here, the order in which the key information generation unit writes the plurality of encrypted media keys to the recording medium may be an order in which the nodes are positioned on the paths from the root to the leaves, the root being a starting point and each node being included only once in the order, and the revocation information generation unit may write the pieces of revocation information to the recording medium in the order. In the user apparatus, the n-ary tree may be composed of a plurality of layers, the order in which the plurality of encrypted media keys are written to the recording medium may be an order in which the nodes are positioned on the paths from the root to the leaves, the root being a starting point, and each node being included only once in the order, the pieces of revocation information may be written to the recording medium in the order, and the specification unit may specify the encrypted media key from amongst the plurality of encrypted media keys written in the order, with use of the plurality of pieces of revocation information written in the order.

7 **GENERATING THE KEY OF THE ROOT NODE STARTING WITH THE KEY OF THE ROOT NODE, USING THE KEY OF A FATHER NODE TO COMPUTE LEVEL BY LEVEL THE KEYS OF ITS CHILD NODES THROUGH TO LEAF NODES**

[0171] The key information generation unit 107 has a variable i that shows the layer number, and a variable j that shows the node name in the layer, the same as the key information header generation unit 106.

[0172] The key information generation unit 107 performs the following processing (a) for each layer excluding the layer 0. In performing the processing (a) for each layer, the variable i showing the layer number holds a value "1", "2", or "3".

Numbered paragraphs [0030] and [0032] again describe the topology of the tree, where the order in which a key information generation unit writes the plurality of encrypted media keys to the recording medium, while numbered paragraphs [0171]-[0182] describes a fact specific algorithm for generating keys, while numbered paragraph [0248] describes obtaining the device key assigned to the root, obtains device keys assigned to nodes whose node names are specific bits of variables, and outputs device keys assigned to each node to the user apparatus. This is a specific teaching of a specific way generating a key, and not a teaching of the claimed subject matter.

[0173] (a) The key information generation unit 107 performs processing (a-1) to (a-3) for each node in the layer whose layer number is shown by the variable i. Here, the name of the node that is the target of processing (a-1) to (a-3) is shown by the variable j.

[0174] (a-1) The key information generation unit 107 obtains the piece of node information that includes the variable j as the node name, from the tree structure table in the tree structure storage unit 102, and judges whether the revocation flag in the obtained node information is "1" or "0".

[0175] (a-2) When the revocation flag is "0", the key information generation unit 107 further judges whether encryption has been performed using the device key that corresponds to the node connected directly above the target node.

[0176] (a-3) When the encryption has not been performed using the device key that corresponds to the node connected directly above the target node, the key information generation unit 107 extracts the device key from the obtained piece of node information, and encrypts the generated media key with use of the extracted device key, by applying an encryption algorithm E1, to generate an encrypted media key.

Encrypted media key=E1(device key, media key)

[0177] Here, E (A, B) shows that data B is encrypted with use of a key A by applying the encryption algorithm E.

[0178] One example of the encryption algorithm E1 is DES (Data Encryption Standard).

[0179] Next, the key information generation unit 107 outputs the generated encrypted media key to the key information recording apparatus 200.

[0180] Note that when the revocation flag is "1", or when encryption has been performed, the key information generation unit 107 does not perform the processing (a-3).

[0181] Since the key information generation unit 107 performs the above-described processing repeatedly as described, in the case shown in FIG. 5, a plurality of encrypted media keys such as those shown in an example in FIG. 7 are generated and output. The key information generation unit 107 outputs the plurality of encrypted media keys as key information D300.

[0182] Note that the positions in which the media keys are stored in the key information D300 are set. These positions are set according to the above-described processing. As shown in FIG. 7, encrypted media keys E1 (keyE, media key), E1(keyG, media key), E1(keyI, media key), E1(keyL, media key) and E1(IK2, media key) a restored respectively in positions defined by "0", "1", "2", "3" and "4".

[8] **GENERATING THE KEY OF THE ROOT NODE STARTING WITH THE KEY OF THE ROOT NODE, USING THE KEY OF A FATHER NODE TO COMPUTE LEVEL BY LEVEL THE KEYS OF ITS CHILD NODES THROUGH TO LEAF NODES .**

[0248] The device key assignment unit 103 obtains the device key assigned to the root (step S222), obtains the device key A assigned to the node whose node name is the head bit of the variable ID (step S223), obtains a device key B assigned to the node whose node name is the head two bits of the variable ID (step S224), obtains a device key C assigned to the node whose node name is the head three bits of the variable ID (step S225), obtains a device key D assigned to the node whose node name is the head four bits of the variable ID (step S226), and outputs the device keys A, B, C, and D assigned to each node to the user apparatus (step S227).

starting with the key of the root node, using the key of a father node to compute level by level the keys of its child nodes through to leaf nodes.

is also said to be taught by the passages at numbered paragraphs [0030], [0032], and [0248]. As noted above, Numbered paragraphs [0030] and [0032] again describe the topology of the tree, where the order in which a key information generation unit writes the plurality of encrypted media keys to the recording medium, while numbered paragraph [0248] describes obtaining the device key assigned to the root, obtaining device keys assigned to nodes whose node names are specific bits of variables, and outputting device keys assigned to each node to the user apparatus. This is a specific teaching of a specific way generating keys, and not a teaching of the claimed subject matter of using the key of a father node to compute level by level the keys of its child nodes through to leaf nodes.

It is conceded in the Office Action that "Nakano does not teach using the key of a father node to compute the child node's keys" but that "McGrew teaches calculating keys from the root to the leaves using a distinct one-way function for each node" citing column 12, lines 14-23[9], and that it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the key of the father node to compute the key of the child node, since McGrew states at column 5, lines 43-48[10] that the key updating method can efficiently generate one or more keys that are applicable to ciphers that are used in multicast and group settings.

[9] In this example embodiment, key updating tree 700 is a binary tree that comprises a root node designated as the Master Key node, a plurality outbound edges 702A, 702B, etc., leading to internal nodes and to leaf nodes. Internal nodes, designated X.sub.2, X.sub.3, etc., lead to leaf nodes designated K.sub.0, K.sub.1, etc. Thus, the root of the tree 700 is associated with a master key, and leaf nodes are associated with keys in a key updating sequence. Each outbound edge 702A, 702B, etc., is associated with a distinct oneway function f.sub.0, f.sub.1. Values of the tree are defined by computing down from the root.

[10] A random access key updating method can efficiently generate one or more future keys in any order. Embodiments are applicable to forward secret ciphers that are used to protect protocols with unreliable transport, and are applicable to ciphers that are used in multicast or other group settings.

What McGrew describes is a random access key updating method to generate one or more future keys in any order.

The amendments have added certain limitations that the combination of Nakano et al with McGrew et al. neither teach nor suggest. Specifically, amended claim 1 recites the following additional limitations:

> 1. (Currently Amended) A method for generating hierarchical keys of digital assets , encrypting the digital assets in a digital asset server, and utilizing the keys of the digital assets and the encrypted digital assets in an associated digital asset client, comprising the steps of:
>
> arranging the digital assets in the digital asset server as at least one tree structure, a root node of the tree structure representing a complete set of the digital assets, other group nodes representing sub-sets in each level of the digital assets respectively, and the nodes in the lowest level being leaf nodes;
>
> randomly generating a key of the root node in the digital asset server; ~~and~~
>
> starting with the key of the root node, using the key of a father node to compute level by level computed keys of its child nodes through to leaf nodes using a one way function, in the digital asset server
>
> encrypting corresponding digital assets in the digital asset server using the computed keys;
>
> requesting an encrypted digital asset at the digital asset client,

determining if a key for the requested encrypted digital asset is present on the digital asset client;

if the digital asset key is not present on the digital asset client, the digital asset client requesting the digital asset key from the digital asset server;

the digital asset server receiver requests from the digital asset client, and thereafter transmits a digital asset key, if requested, and a requested encrypted digital asset from the digital asset server to the associated digital asset client; and

receiving the key and the encrypted digital asset from the digital asset server at the digital asset client and decrypting the encrypted digital asset utilizing the key.

Specifically, the added claim limitations describe an integrated system with complementary server and client, with complementary encryption and decryption, and specifically data structure, metadata, and schema attributes, of:

++encrypting the digital assets in a digital asset server, and utilizing the keys of the digital assets and the encrypted digital assets in an associated digital asset client,

++randomly generating a key of the root node in the digital asset server;

++starting with the key of the root node, using the key of a father node to compute level by level computed keys of its child nodes through to leaf nodes using a one way function, in the digital asset server

++encrypting corresponding digital assets in the digital asset server using the computed keys;

++requesting an encrypted digital asset at the digital asset client, determining if a key for the requested encrypted digital asset is present on the digital asset client;

++if the digital asset key is not present on the digital asset client, the digital asset client requesting the digital asset key from the digital asset server;

++the digital asset server receiver requests from the digital asset client, and thereafter transmits a digital asset key, if requested, and a requested encrypted digital asset from the digital asset server to the associated digital asset client; and

++receiving the key and the encrypted digital asset from the digital asset server at the digital asset client and decrypting the encrypted digital asset utilizing the key.

There is no teaching or suggestion in either or both of Nakano or McGrew of the claimed, total combination of steps

++utilizing the keys of the digital assets and the encrypted digital assets in an associated digital asset client,

++using the key of a father node to compute keys of its child nodes through to leaf nodes using a one way function

++encrypting corresponding digital assets in the digital asset server using the above described computed keys;

++requesting (at the digital asset client) an encrypted digital asset, determining if a key for the requested encrypted digital asset is present on the digital asset client;

++if the digital asset key is not present on the digital asset client, the digital asset client requesting the digital asset key from the digital asset server;

++the digital asset server transmits a digital asset key, if requested, and a requested encrypted digital asset from the digital asset server to the associated digital asset client; and

++receiving the key and the encrypted digital asset at the digital asset client and decrypting the encrypted digital asset utilizing the key.

Neither Nakano et al. nor McGrew et al. disclose client-server computer systems of the type claimed by Applicants, e.g., access to digital assets.

To the contrary, Nakano et al.'s recitation of

> [0218] The decryption unit 404 receives the media key from the decryption unit 403, reads the encrypted content from the recording medium 500c, decrypts the read encrypted content with use of the received media key, by applying a decryption algorithm D2, to generate content, and outputs the generated content to the reproduction unit 405.
>
> Content=D2 (media key, encrypted content)
>
> [0219] Here, the decryption algorithm D2 corresponds to the encryption algorithm E2, and is an algorithm for decrypting data that has been encrypted by applying the encryption algorithm E2

is a more complex system that "receives the media key from the decryption unit, reads the encrypted content from the recording medium…, decrypts the read encrypted content with use of the received media key, by applying a decryption algorithm …, to generate content, and outputs the generated content to the reproduction unit…" and utilizes movable recording media (e.g., elements 500).

This appears to be descriptive of a business model where the decryption and the encrypted, recorded media are separately delivered, and thereafter the media content decrypted by the end user. This is described at numbered paragraph [0009].[11]

Nakano et al. do not recite the attributes of Applicants' claimed client-server system, and McGrew et al describe client-servers in the context of security, but not in the context of providing access to digital assets.

Claim Objections

Claim 11 was objected to and has been rewritten as suggested by the Examiner (paragraph 4).

---

[11] [0009] The key management organization (hereinafter simply referred to as "the organization") has a set of keys that consists of a plurality of device keys and a plurality of media keys. The organization assigns one of the device keys and a device key identification number respectively to each of a plurality of recording apparatuses and a plurality of reproduction apparatuses, and then provides each recording apparatus and reproduction apparatus with the respective device key and device key identification number. In addition, the organization assigns one media key to a recording medium. Next, the organization encrypts the media key, using each of the device keys assigned to the recording apparatuses and the reproduction apparatuses, to generate encrypted media keys, and stores a list of the encrypted media keys corresponding to all the device keys, and the key identification numbers on the recording medium as key information. When the recording medium is loaded into a recording apparatus or a reproduction apparatus, the apparatus extracts the encrypted media key corresponding to the key identification number assigned to the apparatus itself, from the key information in the recording medium, and decrypts the extracted encrypted media key, based on the device key that is assigned to the apparatus itself, to generate the media key. Next, the recording apparatus encrypts content using the obtained media key, and records the resulting encrypted content on the recording medium. On the other hand, the reproduction apparatus decrypts encrypted content in the same way, using the obtained media key. In this way, if a recording apparatus or a reproduction apparatus has a legitimately assigned device key, it is always able to obtain the same media key from the recording medium, thus maintaining compatibility between devices.

Claim Rejections – 35 USC §101

Claims 20-27 were rejected under 35 USC §101 in that the claims recite signals encoded with functional material. Claim 20 has been amended to recite

> "media including a machine-readable data storage medium selected from the group consisting of magnetic hard drives, RAID arrays, RAMACs, a magnetic data storage diskettes, magnetic tape, digital optical tape, RAMs, ROMs, EPROMs, EEPROMs, and flash memories"

to address the rejection under 35 USC §101.

Conclusion

Based on the above discussion, it is respectfully submitted that the pending claims describe an invention that is properly allowable to the Applicants.

If any issues remain unresolved despite the present amendment, the Examiner is requested to telephone Applicants' Attorney at the telephone number shown below to arrange for a telephonic interview before issuing another Office Action.

Applicants would like to take this opportunity to thank the Examiner for a thorough and competent examination and for courtesies extended to Applicants' Attorney.

Certificate of Electronic Filing	Respectfully Submitted
I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being electronically deposited with the United States Patent and Trademark Office on the date shown below.	/s/ Richard M. Goldman
Date of deposit: August 23, 2007	Richard M. Goldman, Reg. # 25,585 371 Elan Village Lane, Suite 208 San Jose, CA 95134
Person mailing paper: Richard M. Goldman	Voice: 408-324-0716
Signature: /s/ Richard M. Goldman	Fax: 408-324-0672 E-mail: goldmanptn@aol.com